

# Selecting the Best Quadrilateral Mesh for Given Planar Shape

Petra Surynková

Johannes Kepler University Linz,  
Altenberger Str. 69, 4040 Linz, Austria  
`petra.surynkova@jku.at`

**Abstract.** The problem of mesh matching is addressed in this work. For a given  $n$ -sided planar region bounded by one loop of  $n$  polylines we are selecting optimal quadrilateral mesh from existing catalogue of meshes. The formulation of matching between planar shape and quadrilateral mesh from the catalogue is based on the problem of finding longest common subsequence (LCS). Theoretical foundation of mesh matching method is provided. Suggested method represents a viable technique for selecting best mesh for planar region and stepping stone for further parametrization of the region.

**Keywords:** quadrilaterals, quadrilateral mesh, optimal mesh, longest common subsequence,  $n$ -sided planar region

## 1 Introduction and Motivation

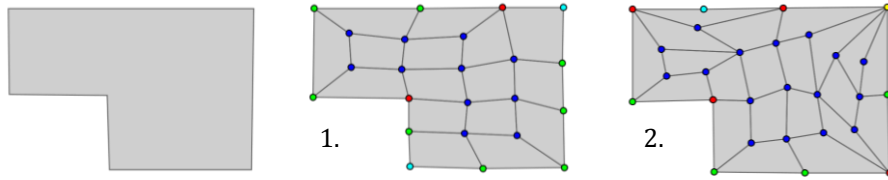
Finding a quadrilateral mesh that matches a given shape - a so called mesh matching problem - represents an important problem in computer aided design, [2], [5], [7], and [11]. The task is specified by user input which describes planar shape to be covered with quadrilateral mesh. In this work we consider only simply connected planar domains which need to be filled by quadrilateral mesh. The planar shape is represented by its boundary drawn in plane typically. In addition to this, the size of mesh, that is, the number of its vertices, is specified by the user as well.

There are many ways how to cover the given shape with a mesh as even single mesh can be matched to the shape in many ways. Moreover there are many non-isomorphic meshes of given number of vertices which makes the problem of finding appropriate mesh for given shape hard. An example of two matchings of different quality between user input and quadrilateral meshes is provided in Figure 1.

Trying to find a suitable matching between the shape and a mesh in an intuitive way by evaluating all the possible coverings would quickly lead to combinatorial explosion as there is exponential number of such covering. We were trying to mitigate this difficulty by finding an appropriate concept from computer science that can be used for reasoning about the mesh matching problem

in a more efficient way. Our finding is that the concept of *longest common subsequence* (LCS) [1], [6] can be employed in mesh matching problem to avoid the combinatorial explosion. Given an objective function for measuring quality of matching, the suitable application of the LCS algorithm would automatically prune out coverings of the input shape by mesh that have no chance to maximize the objective function. Hence instead of dealing with the exponential number of coverings the algorithm goes directly to the optimal one in polynomial number of steps.

In this article we for the first time describe concept of mesh matching to a given planar shape in the formal mathematical way. Having the precise mathematical formulation of the problem we could describe objective function that corresponds to the quality of matching. We also designed a computationally efficient method for mesh selection which is optimal with respect to the matching quality objective. Our method generalizes the common problem of LCS. In contrast to the original LCS, where only the relation of equality between symbols is considered, we allow more general relations between symbols that reflects various cases that arise in mesh matching problem.



**Fig. 1.** Two examples of quadrilateral meshes for user input on the left. The first mesh expresses the shape of user input better than the second mesh. First mesh has lower valences of boundary vertices in *convex* vertices of user input and higher valences in *concave* vertices. The various valences of boundary vertices have different color.

## 2 Background

A quadrilateral mesh, [3], [10], [13] is a triple  $(V, E, Q)$  where  $V$  is a set of vertices,  $E$  is a set of edges, and  $Q$  is a set of quadrilaterals. There exists an embedding of  $(V, E, Q)$  into 2D plane such that each vertex is assigned a point in the plane and each edge is assigned a curve in the plane, so that curves connect vertices and each quad is represented in the plane as a quadrilateral. In our study we furthermore assume only quadrilateral meshes that form a connected, conforming (i.e. free from T-junctions), orientable *2D manifold with boundary*, [3], [9], i.e. our quadrilateral meshes are defined for segmentation of simply connected planar domains.

Regarding the terminology, an edge of the mesh with two incident quads is said to be *internal*, while an edge with just one incident quad is said to be

*boundary*. A vertex of an internal edge is also said to be *internal*, otherwise it is said to be *boundary*. The valence of a vertex is the number of edges incident to that vertex.

## 2.1 Existing Catalogues of Meshes

Our work considers the existing context of literature and software in geometry and computer aided design. Extensive works on providing catalogue of meshes of various sizes had been done previously, [4], [13]. Our approach in this work is to integrate our mesh selection method with existing mesh catalogue that is kept as a database generated by procedural algorithm - that is, our method will select the optimal mesh out of the catalogue of meshes.

The catalogue of meshes which we use as the input for our selection method consists of quadrilateral meshes of certain class and was generated in the previous work, [13]. Without constraints, the number of possible meshes is too high. Thus, we consider only valences  $\leq 5$  for the both boundary and internal vertices which represents standard restriction for numerical applications [8] and what is used in related literature [12], [14].

The quadrilateral meshes in the catalogue furthermore satisfy the following invariant:

- (I) At least one vertex of each internal edge is internal.

Procedural mesh catalogue generates exponential number of meshes with respect to the number of internal vertices of mesh, i.e. the number of internal vertices is specified as the input. Hence, our mesh selection method should make a consideration about matching of a single mesh very quickly to be able to find suitable mesh for given shape in reasonable time. More precisely, we cannot afford to perform any kind of exponential time search or other time consuming operation.

## 3 Longest Common Subsequence Problem

The most simplified version of the longest common subsequence problem consists in finding common subsequence within given two sequences of symbols. Informally said, the task is to delete some symbols in given two sequences so that the resulting sequences - called common subsequences - will be the same. The objective is to obtain as long as possible sequences at the end (in other words we want to make as few as possible symbol deletions) - that is, longest common subsequences.

Consider a simple example of two strings (set of symbols correspond to latin alphabet) "alpha" and "aleph". These two sequences are obviously different but after deleting last 'a' in the first sequence and middle 'e' in the second string we obtain "alph" in both cases which is the longest common subsequence for this example.

The very positive aspect about the LCS problem is that a variety of efficient (polynomial time) algorithms exist that solve this optimization problem, [15], [16].

### 3.1 Application in Shape Matching

Low time complexity makes LCS algorithms good candidates for using them as a basis for consideration about mesh matching. However, mesh matching problem and LCS problem are completely different concepts hence we need to show first what are the similarities between these problems.

When a user specifies his planar shape we can regard his input as an abstract information that can be annotated by a sequence of symbols. Information about the *length* of edges of the boundary, which vertices on the boundary are *convex*, which are *concave* can be read from the input. Such information can be encoded into a sequence of symbols.

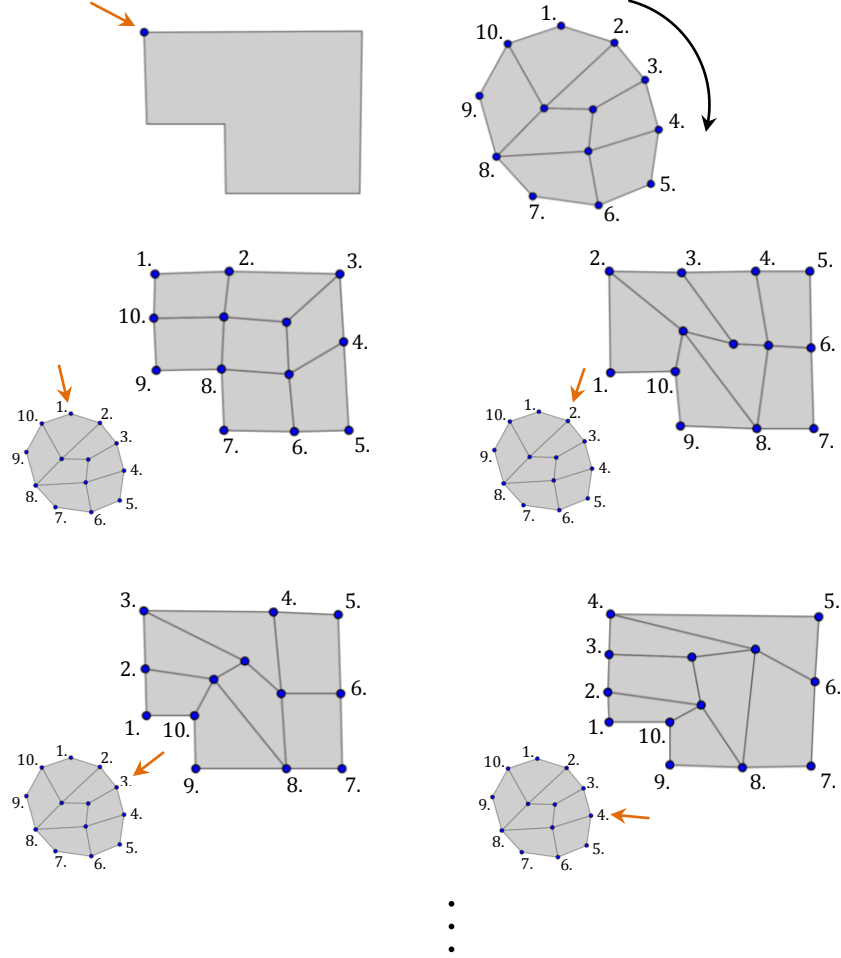
At the same time, we need to annotate boundaries of meshes stored in the catalogue using a sequence of symbols in correspondence with annotation of inputs. There is considerable discrepancy between user inputs in the form of planar shapes and representation of meshes in existing catalogues - most catalogues represent meshes in the abstract form as list of quadrilaterals and their interconnections. Moreover we need to reflect certain level of flexibility of mesh matching with respect to given shape - a mesh may be matched to the shape in a not ideal way while no better matching is possible.

We therefore introduce *lattice* of classification of mesh boundary vertices with respect to properties of *convexity*, *concavity*, and *others*. Unlike in the case of standard LCS where we compare a pair of symbols whether they are same or not, here we are more flexible. When we compare a vertex from user input shape with a boundary vertex of mesh from the catalogue, the quality of matching between these two vertices is determined by the *lattice*. For example if a *convex* point from user input is being compared with a vertex from mesh, then *lattice* for classification of *convex* vertices is used to determine the quality of correspondence between the two (the *lattice* thus provide classification from complete match between *convex* and *convex* point to complete mismatch between *convex* and *concave* point).

This operation of comparison between a point from user input and a boundary vertex of a mesh comes as a parameter to an LCS algorithm instead of standard equality between symbols. The output of the algorithm hence will be pair of sequences whose symbols at corresponding positions represent best possible match according to the *lattice*. When this output is interpreted back to the world of geometry we have an optimal matching of a given mesh from catalogue to the given user input. Hence we are able to select optimal mesh from the catalogue provided that consideration about the single mesh is fast enough.

The method works for fixed starting point in a testing quadrilateral mesh which is compared to fixed point in the given user input. For finding an optimal matching between the user input and a mesh from the catalogue we consider

all rotation of a testing mesh, i.e. for fixed point in the user input we test all possible starting points in a mesh. This approach is illustrated in Figure 2.



**Fig. 2.** An example of mesh matching between one quadrilateral mesh from the catalogue and user input. Point in the user input is fixed and the rotations of mesh are considered.

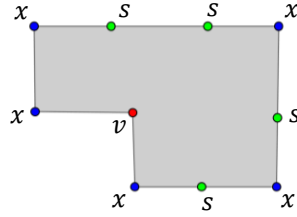
## 4 Optimal Mesh Selection Method

Our basic mesh matching method assumes a catalogue of meshes represented as a list of interconnected quadrilaterals. The user input given as a planar shape is

further processed and is assigned a sequence of boundary points. We distinguish three types of boundary vertices, see Figure 3:

- **straight** point - denoted by symbol  $s$
- **convex** point - denoted by symbol  $x$
- **concave** point - denoted by symbol  $v$

*Convex* and *concave* points are extracted naturally from the user defined input planar shape. *Straight* points are assigned to lines of the boundary of shape. That is, each line is assigned certain number of internal *straight* points according to its length. Longer boundary line have more internal *straight* vertices.



**Fig. 3.** Three types of boundary vertices in the given user input.

As meshes in the catalogue are represented in the abstract way there is not much information available from which a corresponding annotation can be constructed. The only usable information about meshes are valences of their boundary vertices which in case of quadrilateral meshes are from the range 2, 3, 4, 5. To further increase amount of information about a given boundary vertex we also consider valences of its neighbors which allows us to determine which vertex is more likely *convex* or which vertex is more likely *concave*.

Intuitively *convex* boundary vertices have low valence while their neighbors have high valence. To be able to formalize this likeliness to be *straight*, *convex*, or *concave* point we introduce a lattice for each type of point. Figure 5 shows a suggested lattice for *convex* point in the given user input. Some triples of valences of boundary points in the lattice are weeded out because they cannot occur in our certain class of meshes. The lattices for *concave* and *straight* points are designed analogically.

The higher the vertex is classified within the lattice the more likely it can be matched to a *convex* point on the user input. The design of lattice is a matter of careful consideration of visual appearance of matching and experience of the expert designer. Furthermore, each level of lattice is assigned an integer weight that will be reflected by the modified LCS algorithm when comparing symbol from the input sequence with mesh boundary vertex. Positive weights represent likeliness of match between points while negative values stand for likeliness of mismatch. Absolute value of the weight represent measure of match or mismatch.

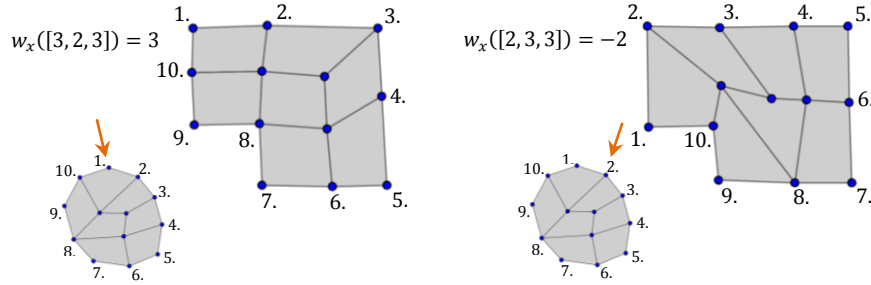
A special weight  $-\infty$  is reserved to denote complete mismatch between the pair of vertices (this corresponds to disequality between symbols in the standard LCS algorithm).

Our modified LCS algorithm assumes two input sequences of symbols - the first from user input; the second obtained by annotating mesh from the catalogue. The symbols of the second sequence have the following format of triples:  $[u, v, w]$  where  $v \in 2, 3, 4, 5$  is a valence of mesh boundary vertex, and  $u, w \in 2, 3, 4, 5$  are valences of counter-clock-wise and clock-wise neighbors of  $v$  respectively.

Weights are assigned to triples  $[u, v, w]$  by weight functions. Formally we introduce a weight function for each type of point. That is, we have functions:

$$w_s, w_x, w_v : \{2, 3, 4, 5\}^3 \rightarrow \mathbb{Z} \cup \{-\infty\}$$

which assigns triples of valences their weights with respect to the lattice for *straight*, *convex*, and *concave* point, respectively, see an example in Figure 4. These weight functions are used by the modified LCS algorithm when it is making comparing a pair of symbols (first one from the first sequence second one from the second sequence).



**Fig. 4.** An example of weights of selected mesh boundary vertex with respect to the fine-grained lattice for *convex* point.

We omit implementation details of the modified LCS algorithm for the sake of brevity. However, important high level specialty of our version of LCS is that it assumes the length of the first sequence to be at least the length of the second sequence. Moreover, deletions of symbols can be made from the first sequence only (from the user input). It is natural assumption as we want to match all the vertices of the catalogue mesh to some point in the user defined shape.

The two sequences of the same length are valued by a utility which is calculated as the sum of weights of symbols from the second sequence with respect to lattices corresponding to respective symbols from the first sequence. The task of our modified LCS algorithm is to compute longest common subsequence out of given user defined input sequence (the first sequence) with the highest possible utility.

Formally we define overall utility function of a pair of sequences of symbols discussed above as follows

$$f([s_1, \dots, s_n], [[u_1, v_1, w_1], \dots, [u_n, v_n, w_n]]) = \sum_{i=1}^n w(s_i, [u_i, v_i, w_i])$$

where

$$w(s_i, [u_i, v_i, w_i]) = \begin{cases} w_s([u_i, v_i, w_i]), & \text{if } s_i = \mathbf{s} \\ w_x([u_i, v_i, w_i]), & \text{if } s_i = \mathbf{x} \\ w_v([u_i, v_i, w_i]) & \text{if } s_i = \mathbf{v} \end{cases}$$

The algorithm finds a sequence  $[s_1, \dots, s_n]$  for the input pair of sequences  $[s'_1, \dots, s'_m]$  and  $[[u_1, v_1, w_1], \dots, [u_n, v_n, w_n]]$  where  $n \leq m$  so that  $[s_1, \dots, s_n]$  is a subsequence of  $[s'_1, \dots, s'_m]$  and its  $f$  value is maximum.

#### 4.1 Design of Lattices

The design of *lattice* is a matter of experience of the expert designer. We suggested a lattice for each type of point in the user input. Our lattices are fine-grained so they enable to evaluate mesh matching more precisely than coarse lattices. An example of coarse lattice for *convex* point is shown in Figure 6 and an example of coarse lattice for *straight* point is shown in Figure 7.

Note that occurrence of  $-\infty$  represents strict impossibility to match a point of given type from user input (corresponds to the type of lattice) to a vertex from a mesh. If the LCS algorithm cannot find a correspondence between point from the user input and some mesh vertex so that utility other than  $-\infty$  is assigned to that correspondence the point must be treated as deleted by the LCS algorithm. This is the only case when LCS performs deletion from some of its input sequences.

#### 4.2 Properties of the Method

Clearly as LCS algorithms are optimal in their nature our modified algorithm enables finding optimal match between used defined shape and mesh from the catalogue with respect to given objective function optimally. Moreover the algorithm requires polynomial time and space which makes it an excellent candidate for selecting a mesh with the best match for the given user input.

### 5 Conclusions and Future Work

A new approach for finding a quadrilateral mesh from the catalogue of quadrilateral meshes of certain class that matches a given user input was described in this work. The method conceptually builds on the known problem of finding longest common subsequence (LCS). As LCS and mesh matching are fundamentally different problems we proposed a series of techniques that allow us to transform



mesh matching problem to LCS. These techniques include adaptation of the LCS algorithm and introduction of symbol comparison based on lattices that model likelihood of correspondence between user input points and mesh vertices.

The theoretical foundation of a method is provided. The major contribution is that viewing mesh matching problem through the concept of LCS allows mitigating the combinatorial complexity. Instead of evaluating all possible matchings between the user input and a mesh from the catalogue in the exponential time, the adapted LCS algorithm rules out partial matchings as early as possible if they turn out not to be optimal which leads eventually to the polynomial time.

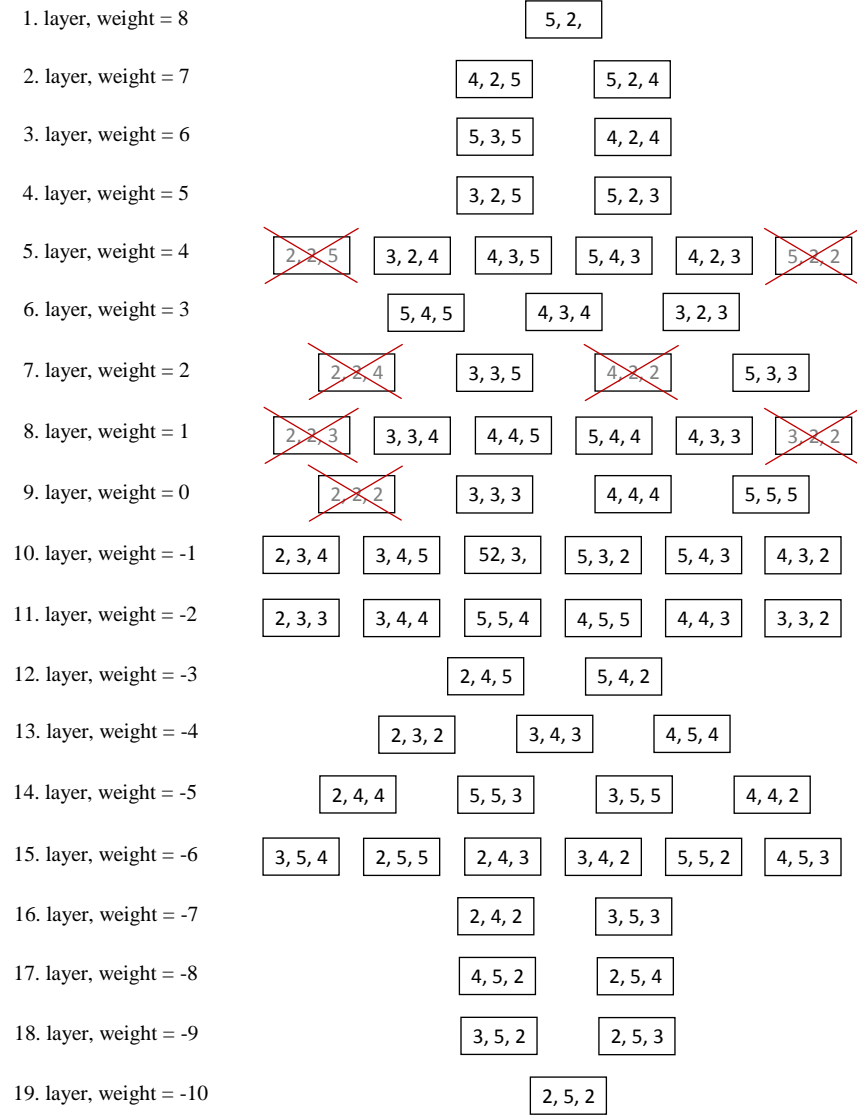
In the future work we will focus on experimental evaluation which will be targeted on visual comparison of match matching obtained using fine-grained and coarse lattices. The interconnection of the mesh matching algorithm and the procedural catalogue is planned too.

Another interesting topic for the future work is to develop new techniques for evaluation best mesh according to the distribution of internal vertices.

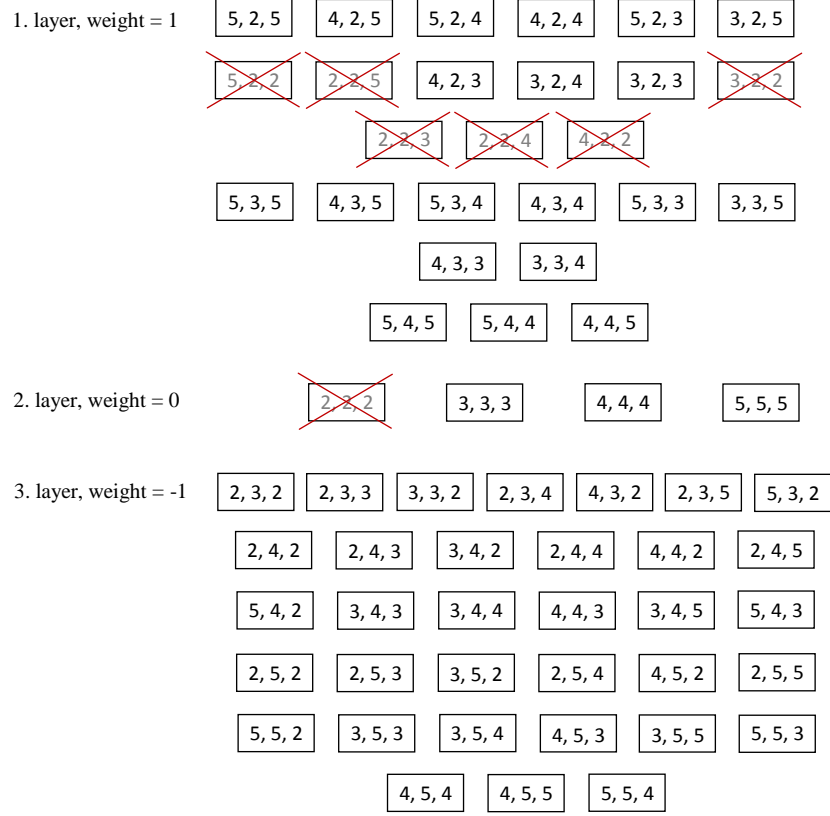
## References

1. Bergroth, L., Hakonen, H., Raita, T.: A survey of longest common subsequence algorithms. In: Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00). pp. 39–48. IEEE Computer Society, Washington, DC, USA (2000)
2. Bommès, D., Lévy, B., Pietroni, N., Puppo, E., Silva, C., Tarini, M., Zorin, D.: Quad-mesh generation and processing: a survey. *Computer Graphics Forum* 32(6) (2013)
3. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B.: *Polygon Mesh Processing*. A K Peters, Natick (2010)
4. Buchegger, F., Jüttler, B.: Planar multi-patch domain parametrization via patch adjacency graphs. *Computer-Aided Design* (2016 in press)
5. Edelsbrunner, H.: *Geometry and Topology for Mesh Generation*. Cambridge University Press, New York (2001)
6. Hirschberg, D.S.: Algorithms for the longest common subsequence problem. *J. ACM* 24(4), 664–675 (1977)
7. Kälberer, F., Nieser, M., Polthier, K.: Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum* 26(3) (2007)
8. Liu, Y., Xing, H.L., Guan, Z.: An indirect approach for automatic generation of quadrilateral meshes with arbitrary line constraints. *Numerical Methods in Engineering* 87(9) (2011)
9. Luebke, D., Reddy, M., Cohen, J.D., Varshney, A., Watson, B., Huebner, R.: *Level of Detail for 3D Graphics*. Morgan Kaufmann Publishers, San Francisco (2003)
10. Mitchell, S.A.: A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume. In: 13th Annual Symposium on Theoretical Aspects of Computer Science (1996)
11. Nasri, A., Sabin, M., Yasseen, Z.: Fillingn-sided regions by quad meshes for subdivision surfaces. *Computer Graphics Forum* 28(6), 1644–1658 (2009)
12. Peng, C.H., Barton, M., Jiang, C., Wonka, P.: Exploring quadrangulations. *ACM Transactions on Graphics* 33(1) (2014)

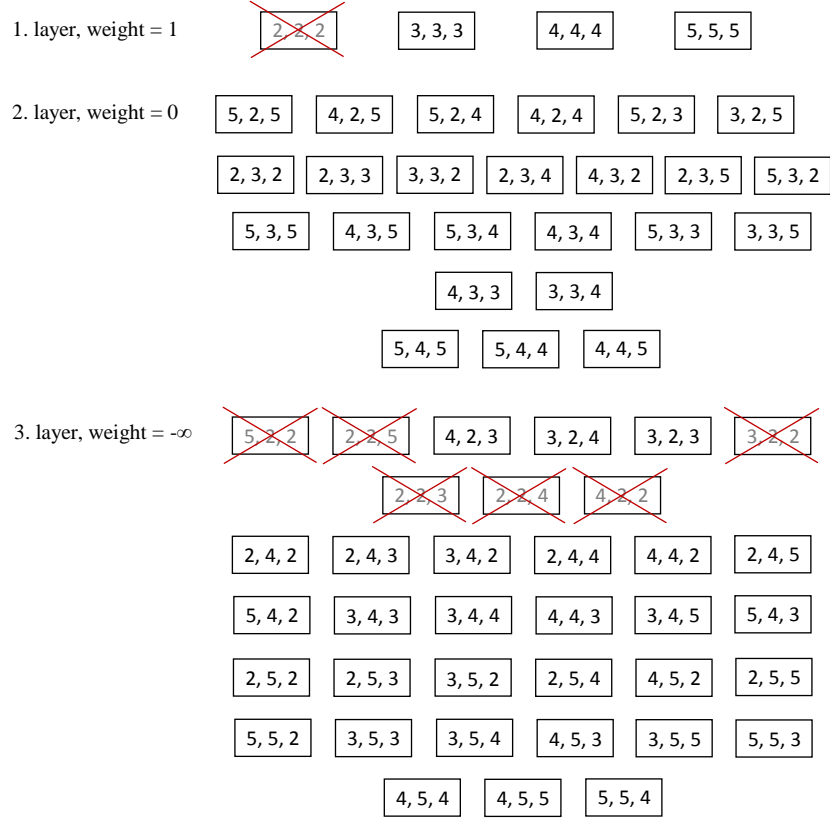
13. Surynkova, P., Jüttler, B., Buchegger, F., Surynek, P.: Enumerating quadrilateral meshes. Technical report no. 51, NFN Geometry and Simulation, Johannes Kepler University Linz (2016)
14. Takayama, K., Panozzo, D., Sorkine-Hornung, O.: Pattern-based quadrangulation for  $N$ -sided patches. *Computer Graphics Forum* 33(5) (2014)
15. Ukkonen, E.: Algorithms for approximate string matching. *Inf. Control* 64(1-3), 100–118 (1985)
16. Ullman, J.D., Aho, A.V., Hirschberg, D.S.: Bounds on the complexity of the longest common subsequence problem. *J. ACM* 23(1), 1–12 (1976)



**Fig. 5.** Suggested lattice for *convex* point in the given user input.



**Fig. 6.** An example of coarse lattice for *convex* point in the given user input.



**Fig. 7.** An example of coarse lattice for *straight* point in the given user input.